VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Metody rozdělení grafu a jejich aplikace na řešení vybraných inženýrských úloh

Graph partitioning methods and their application on the solution of selected engineering problems

2010 Jan Saferna

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

V Ostravě 23. července 2010 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 23. července 2010 .....

Na tomto místě bych chtěl poděkovat Doc. Ing. Tomáši Kozubkovi Ph.D. a Ing. Pavle Kabelíkové za cenné rady, podněty a připomínky při tvorbě diplomové práce. Dále M.Sc. Shehzadovi Afzalovi za nápomoc během mojí stáže na Montanuniversität v Leobenu. V neposlední řadě své rodině za podporu během mých studijních let.

Abstrakt

Tato diplomová práce se zabývá metodami rozkladu grafu na podgrafy, jež jsou souvislé. Jsou zde popsány metody rozkladu založené na grafových algoritmech, které jsou jednoduché, a metody založené na hlubších znalostech teorie grafů a lineární algebry, jež jsou rychlé a efektivní. Tyto rozklady grafu nachází uplatnění v nejrůznějších technických úlohách.

Klíčová slova: rozklad grafu, pseudo-periferiální vrchol, prohledávání do šířky, prohledávání do hloubky, Lanczosova metoda

Abstract

In the present work we study graph partitioning methods. We describe easy methods based on graph algorithms as well methods based on deep knowledge of graph theory and linear algebra. These methods are fast and effective.

Keywords: graph partitioning, pseudo-peripheral node, breadth-first search, depth-first search, Lanczos method

Seznam použitých zkratek a symbolů

AND	–	logická spojka *„a současně“*
BFS	–	prohledávání do šířky
DFS	–	prohledávání do hloubky
FETI	–	finite element tearing and interconnect
MLSE	–	víceúrovňová expanze nastavení vrcholů
RGB	–	rekurzivní grafová bisekce

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Rozdělení grafu v poslední době nachází aplikaci v efektivním řešení nejrůznějších úloh. Nejvýznamnější aplikací rozdělení grafu je využití při paralelní a distribuované realizaci různých algoritmů. Princip řešení spočívá v rozdělení velké úlohy na několik menších, které se navzájem minimálně ovlivňují. V práci je popsána dekompozice grafu založená na grafových algoritmech, jež činí rozklad jednoduchý, rychlý a efektivní. Dekompozice je založena na velmi dobrých znalostech lineární algebry a teorie grafů; je tedy náročná na znalosti, na druhou stranu je extrémně rychlá, efektivní a použitelná pro velké úlohy. Také je zde popsána víceúrovňová grafová bisekce, která pomocí převodu grafu na graf s výrazně nižším počtem vrcholů řeší velké úlohy extrémně rychle a velmi efektivně. Aplikace dekompozice grafu nachází uplatnění v různých numerických metodách, například v metodě konečných prvků, FETI nebo total FETI. Algoritmy jsem implementoval v matlabu viz příloha.

Tato práce je rozdělena do následujících kapitol. Druhá kapitola je věnována definicím a základním větám, na kterých stojí tato teorie rozkladu grafu na oblasti. Ve třetí kapitole jsou popsány algoritmy rozdělující graf na oblasti. Čtvrtá kapitola se zabývá nalezením centra grafu. V páté kapitole jsou popsány úpravy vzniklých oblastí, které zlepší rozdělení grafu. Šestá kapitola popisuje víceúrovňovou grafovou bisekci, která daný graf převede na menší odpovídající graf, na kterém je provedeno rozdělení na oblasti. V sedmé kapitole aplikujeme získané poznatky na inženýrské úlohy. V osmé kapitole jsou popsány metody založené na shlukování vrcholů.

2 Základní pojmy a algoritmy z teorie grafů

2.1 Síť a oblast sítě

Tato práce se zabývá rozdělením sítě na oblasti. Pojmy síť, oblast a rozdělení sítě na oblasti jsou představeny v následujících definicích. Z důvodu praktických potřeb je síť definována v prostorech $\mathbb{R}^2$ a $\mathbb{R}^3$. V prostoru $\mathbb{R}^3$ je zapotřebí rozlišit diskretizaci na dvojrozměrné a trojrozměrné objekty. Pro zjednodušení nazveme síť definovanou v prostoru $\mathbb{R}^2$ diskretizační sítí obsahu, síť definovanou v prostoru $\mathbb{R}^3$ s dvojrozměrnými objekty diskretizační sítí povrchu a síť definovanou v prostoru $\mathbb{R}^3$ s trojrozměrnými objekty diskretizační sítí objemu. Grafem se rozumí uspořádaná dvojice $G = (V, E)$, kde V je neprázdná konečná množina vrcholů a E je konečná množina dvouprvkových podmnožin s různými prvky množiny V. Prvky množiny E se nazývají hrany. Podgraf H grafu G je graf splňující $V(H) \subseteq V(G)$ a $E(H) \subseteq E(G)$. Indukovaný podgraf H grafu G je podgraf splňující $E(H) = E(G) \cap \binom{V(H)}{2}$. Planární graf je graf, který je možné nakreslit v euklidovské rovině tak, aby dvě libovolné hrany neměly jiné společné body, než koncové. Rovinný graf je graf, jenž je nakreslen v euklidovské rovině tak, že žádné dvě hrany se nekříží a zároveň každá hrana obsahuje právě dva vrcholy. Prostorový graf je graf, jehož nakreslení v prostoru $\mathbb{R}^3$ neobsahuje protnutí dvou hran ani protnutí hrany s vrcholem, který není jejím koncovým vrcholem. Je-li n přirozené a zobrazení $\gamma : \langle 0; 1\rangle \to \mathbb{R}^n$ prosté a spojité, pak křivka γ je podmnožina prostoru $\mathbb{R}^n$ ve tvaru $k = \gamma(\langle 0; 1\rangle) = \{\gamma(t) : t \in \langle 0; 1\rangle\}$. Body $\gamma(0)$ a $\gamma(1)$ se nazývají koncové body křivky γ. Jordanova křivka je křivka, jejíž koncové vrcholy splývají.

Věta 2.1 *Každá Jordanova křivka γ: $\langle 0, 1\rangle \to \mathbb{R}^2$ rozděluje rovinu na právě dvě souvislé části, vnitřek a vnějšek této křivky. Jordanova křivka γ je společnou hranicí těchto oblastí.*

Následující definice převzatá z [16] zavádí stěny grafu.

Definice 2.1 *Nechť $A \subseteq \mathbb{R}^n$ je nějaká podmnožina roviny. Nazveme ji souvislou, pokud pro $\forall x, y \in A$ platí, že existuje křivka o s koncovými body x a y takovými, že $o \subseteq A$. Křivky příslušné hranám nějakého rovinného grafu pak podle relace souvislosti rozdělují rovinu na třídy ekvivalence, které se nazývají stěny grafu.*

Stěna, která je v rovinném nakreslení grafu neomezená, se nazývá vnější stěnou, ostatní jsou stěny vnitřní. Hrana, která odděluje vnější stěnu od vnitřní stěny, se nazývá vnější hrana. Hrana oddělující dvě vnitřní stěny se nazývá hranou vnitřní. V prostoru $\mathbb{R}^3$ je situace složitější.

Definice 2.2 *Nakreslením grafu $G = (V, E)$ se rozumí zobrazení $G \to \mathbb{R}^n$, které vrcholům grafu přiřadí různé body prostoru a hranám E přiřadí úsečku s koncovými body, na které jsou zobrazeny koncové vrcholy příslušné hrany. Jestliže je oborem hodnot zobrazení nějaká podmnožina $\mathbb{R}^2$, je zobrazení rovinné. V případě, že je obor hodnot podmnožinou $\mathbb{R}^3$, se zobrazení nazývá prostorové.*

Základním kamenem přechodu od grafu k síti je hranice stěny. Pro jednoduchost je hranice stěny omezena na maximální délku 4.

Definice 2.3 *Hranicí stěny v grafu G se rozumí podgraf C_3 nebo indukovaný podgraf C_4.*

Potřebujeme, aby nakreslení všech vrcholů hranice stěny určovalo právě jednu rovinu. Nyní zavedeme pojem stěny grafu.

Definice 2.4 *Nechť je dán prostorový graf G. Stěnou se rozumí konvexní obal jakékoli hranice stěny v G.*

Nyní zavedeme důležitý pojem elementu.

Definice 2.5 *Elementem obsahu se rozumí jakákoli omezená stěna příslušného grafu pro diskretizační sítě obsahu a povrchu.*

Definice 2.6 *Nechť je dán graf G s nakreslením v prostoru $\mathbb{R}^3$, jenž reprezentuje diskretizaci objemu. Odebráním všech stěn, hran a vrcholů grafu G z prostoru $\mathbb{R}^3$ se tento prostor rozpadne na několik souvislých množin. Každá tato omezená množina se nazývá elementem objemu.*

Nyní se podívejme na definici pojmu síť.

Definice 2.7 *Nechť $G = (V, E)$ je graf. Sítí se nazývá dvojice $M = (V, K)$, kde V je množina vrcholů a jejich pozice v prostoru a K je množina jednoduchých celků složený z vrcholů, hran a případně i stěn, které určují element sítě grafu G.*

Nyní zavedeme rozdělení grafu na oblasti.

Definice 2.8 *Rozdělením grafu G na oblasti se rozumí nalezení podgrafů $G_1, \ldots, G_n$ sítě G, které splňují následující vztahy:*

- $\bigcup_{i=1}^{n} V(G_i) = V(G)$,
- $V(G_i) \cap V(G_j) = \emptyset, \qquad$ *pro* $i \neq j$.

Libovolný podgraf G_i z tohoto rozdělení se nazývá oblast grafu G.

2.2 Nejkratší cesta

Základním pojmem všech grafových algoritmů je cesta, což je jakákoli posloupnost vrcholů $P = (p_0, p_1, \ldots, p_n)$, jež splňuje následující podmínky:

- $\forall i \in \{0, 1, \ldots, n\}, \forall j \in \{0, 1, \ldots, n\}, i \neq j : p_i \neq p_j$,
- $\forall i \in \{0, 1, \ldots, n-1\} : (p_i, p_{i+1}) \in E(G)$.

Z potřeby měřitelnosti cesty a následné porovnatelnosti jednotlivých cest se zavádí délka cesty, což je číslo ohodnocující danou cestu.

Definice 2.9 *Nechť G je graf a $w\colon E(G) \to \mathbb{R}^+$ je libovolná funkce. Pod pojmem délky cesty $P = (p_0, p_1, \ldots, p_n)$ rozumíme hodnotu $|P| = \sum_{i=0}^{n-1} w(p_i p_{i+1})$.*

Pro hranově ohodnocený graf nazýváme $w(e)$ váhou hrany. Jestliže není graf váhově ohodnocený, klademe $w \equiv 1$.

Následující dvě definice zavádějí dva užitečné pojmy, které se velmi často využívají. Jedná se o minimální cestu a vzdálenost mezi vrcholy.

Definice 2.10 *Cesta, která z vrcholu u do vrcholu v ze všech cest mezi těmito vrcholy nabývá nejmenší délky, je minimální cesta.*

Definice 2.11 *Vzdálenost vrcholu u od jiného vrcholu v je délka nejkratší cesty z vrcholu u do vrcholu v a značí se $dist(u, v)$. Jestliže z vrcholu u do vrcholu v neexistuje žádná cesta, klademe $dist(u, v) = \infty$.*

Jedním z nejrozšířenějších algoritmů pro nalezení nejkratší cesty je Dijkstrův algoritmus a je popsán v [11]. Složitost tohoto algoritmu je $O(|E| + |V|log|V|)$.

2.3 Excentricita, poloměr a průměr grafu

Při řešení různých problémů často potřebujeme znát kvalitativní údaje o vzdálenostech dvou vrcholů v grafu. S odhadem těchto hodnot nám pomáhají grafové parametry, jež zavedeme v následujících definicích.

Definice 2.12 *Excentricita vrcholu v je značena $ecc(v)$, kde $ecc(v) = \max_{u \in V(G)} \{dist(u, v)\}$.*

Definice 2.13 *Průměr grafu G je značen $diam(G)$ a definován přepisem*
$diam(G) = \max_{v \in V(G)} \max_{u \in V(G)} \{dist(u, v)\}$.

Definice 2.14 *Poloměr grafu G je značen $rad(G)$ a definován přepisem*
$rad(G) = \min_{v \in V(G)} \max_{u \in V(G)} \{dist(u, v)\}$.

2.4 Prohledávací algoritmy

Prohledávací algoritmy patří mezi základní grafové algoritmy. Tyto algoritmy postupně prohledají všechny z výchozího vrcholu dostupné vrcholy grafu. Výhodou těchto algoritmů je jednoduchost a široká aplikovatelnost. V mnoha konkrétních aplikacích však existují rychlejší algoritmy na řešení dané úlohy.

2.4.1 Prohledávání do hloubky

Prohledávání do hloubky patří mezi základní prohledávací algoritmy. Tento algoritmus charakterizuje výběr vrcholu, který bude následně zpracován. Algoritmus upřednostňuje později objevené vrcholy. Vrcholy, které byly dosud objevené[1], ale ještě nezpracované,

[1] Vrcholy, jejichž alespoň jeden sousední vrchol už byl zpracován.

jsou uloženy v zásobníku. Výhodou tohoto algoritmu jsou nízké paměťové nároky, nevýhodou velké časové nároky. Při hledání jednoho z vhodných vrcholů je časová náročnost velmi ovlivněna pořadím v jakém jsou následující vrcholy vybraného vrcholu uspořádány.

Algoritmus prohledávání do hloubky zapsaný v pseudokódu:

1. Všem vrcholům x nastav stav VISIT(x) = FALSE.
2. Počátečnímu vrcholu s nastav stav VISIT(s) = TRUE.
3. Vlož počáteční vrchol s na zásobník.
4. Vezmi uzel z vrcholu zásobníku a označ jej u.
5. Každý uzel v, do kterého vede hrana z vrcholu u a jeho stav VISIT(v) je roven FALSE, vlož na zásobník a změň jeho stav VISIT(v) = TRUE.
6. Pokud zásobník není prázdný, opakuj od bodu 4.

Asymptotická složitost tohoto algoritmu je $O(|V| + |E|)$.

2.4.2 Prohledávání do šířky

Druhým základním prohledávacím algoritmem je prohledávání do šířky. Také tento algoritmus charakterizuje výběr vrcholu, který bude následně zpracován. Na rozdíl od prohledávání do hloubky tento algoritmus upřednostňuje dříve objevené vrcholy. Vrcholy, které byly dosud objevené[2], ale ještě nezpracované, jsou uloženy ve frontě. Výhodou tohoto algoritmu jsou nízké časové nároky, nevýhodou velké paměťové nároky. Časová náročnost není ovlivněna pořadím, v jakém jsou následující vrcholy vybraného vrcholu uspořádány.

Algoritmus prohledávání do šířky zapsaný v pseudokódu:

1. Všem vrcholům x nastav stav VISIT(x) = FALSE.
2. Počátečnímu vrcholu s nastav stav VISIT(s) = TRUE.
3. Vlož počáteční vrchol s do fronty.
4. Vezmi první vrchol z fronty a označ jej u.
5. Každý vrchol v, do kterého vede hrana z uzlu u a jeho stav VISIT(v) je roven FALSE, přidej na konec fronty a změň jeho stav VISIT(v) = TRUE.
6. Pokud fronta není prázdná, opakuj od bodu 4.

Asymptotická složitost algoritmu je $O(|V| + |E|)$.

[2] Vrcholy, jejichž alespoň jeden sousední vrchol už byl zpracován.

2.4.3 Paralelní prohledávaní do šířky

Prohledávání do šířky má i svoji paralelní verzi, která se uplatní hlavně při dělení grafu. Tento algoritmus na rozdíl od základní verze prohledávání do šířky vyžaduje dva výchozí vrcholy. Pracuje se dvěma frontami, každý vrchol má dva stavy na sobě nezávislé. Algoritmus navíc potřebuje znát vzdálenost každého vrcholu od obou výchozích vrcholů.

Algoritmus paralelního prohledávání do šířky zapsaný v pseudokódu:

1. Vstupem je graf G a počáteční vrcholy x_1 a x_2.
2. Všem vrcholům v z $V(G)$ nastav stavy 1 a 2 vrcholu v na NEOBJEVEN.
3. Počátečnímu vrcholu x_1 nastav stav 1 na OBJEVEN, nastav vzdálenost 1 na 0 a vlož ho do fronty číslo 1.
4. Počátečnímu vrcholu x_2 nastav stav 2 na OBJEVEN, nastav vzdálenost 2 na 0 a vlož ho do fronty číslo 2.
5. Vyber frontu, jejíž první vrchol nabývá nejnižší příslušné vzdálenosti a označ ji b. Jestliže byla vybrána fronta jedna, tak nastav index i na 1, v případě výběru fronty 2 nastav index i na 2.
6. Vezmi vrchol z fronty b a označ jej u. Změn stav i vrcholu u na PRIRAZEN.
7. Pro každý vrchol v, do kterého vede hrana z vrcholu u a jeho stav i je NEOBJEVEN, nastav jeho vzdálenost i o 1 větší než je vzdálenost i vrcholu b, nastav jeho stav i na OBJEVEN a vlož ho do fronty b.
8. Pokud obě fronty nejsou prázdné, opakuj od bodu 5.

Asymptotická složitost algoritmu je $O(|V| + 2 * |E|)$.

Věta 2.2 *Nechť je G souvislý graf, $u, v \in V(G), u \neq v$. Nechť G_u a G_v jsou oblasti vzniklé paralelním prohledávání grafu G do šířky. Pak oblasti G_u a G_v jsou souvislé.*

Důkaz. Nechť $\exists x \in G_u : dist_{G_u}(u, x) = \infty$. Ze souvislosti G plyne, že $dist_G(u, x) < \infty$. Proto všechny cesty z vrcholu u do vrcholu v obsahují alespoň jeden vrchol nepatřící G_u. Algoritmus paralelní BFS pracuje následovně. V inicializaci vloží do fronty F_u vrchol u a do fronty F_v vloží vrchol v.

Dokud obě fronty nejsou prázdné, vyber frontu, jejíž první vrchol má menší vzdálenost, z ní vyber vrchol a vlož do fronty všechny sousední vrcholy posledně vybraného vrcholu, které ještě ve frontě nebyly. Všem právě vloženým vrcholům do fronty nastav vzdálenost o 1 větší než je vzdálenost posledně vybraného vrcholu.

Vrchol k je přiřazen oblasti G_u pouze tehdy, je-li vybrán z fronty F_u.

Jestliže byl vrchol b následně přiřazen oblasti G_u, byl krok před přiřazením vybrán z fronty F_u, proto musel být do fronty F_u vložen, tedy $\exists a \in V(G_u) : ab \in E(G)$ $AND\ dist_{G_u}(u,a) < \infty$. A proto $dist_{G_u}(u,b) < \infty$. ■

2.5 Pseudo-periferiální vrchol

Při řešení úloh je výhodné zvolit výchozí vrchol, který se nachází na okraji grafu. Vrcholy, které leží skutečně na periferii grafu se nazývají periferiální.

Definice 2.15 *Vrchol $v \in V(G)$ se nazývá periferiální, jestliže $ecc(v) = diam(G)$.*

Nevýhodou periferiálního vrcholu při realizaci řešení problému je vysoká časová náročnost algoritmu, který ho nalezne. Proto je výhodnější nehledat vrchol, který je nejvíce na okraji grafu, ale spokojit se s vrcholem, jenž je dostatečně na okraji a je rychle nalezitelný, což splňuje pseudo-periferiální vrchol.

Definice 2.16 *Vrchol $v \in V(G)$ se nazývá pseudo-periferiální, jestliže $\exists u \in V(G) : dist(v,u) = ecc(u) = ecc(v)$.*

Definice 2.17 *Nechť je dán graf G a pseudo-periferiální vrchol $u \in V(G)$. Vrchol $v \in V(G)$ je alternativní pseudo-periferiální vrchol k vrcholu u[3], jestliže $dist(v,u) = ecc(u) = ecc(v)$.*

Algoritmus nalezení pseudo-periferiálního vrcholu zapsaný v pseudokódu:

1. Vyber inicializační vrchol x a nastav $d := 0$.
2. Nalezni nejvzdálenější vrchol z vrcholu x a označ ho y.
3. $d := dist(x,y)$
4. $x := y$
5. Nalezni nejvzdálenější vrchol z vrcholu x a označ ho y.
6. Pokud $dist(x,y) > d$, opakuj od bodu 3.
7. Vrchol x je pseudoperiferiální vrchol a vrchol y je alternativním pseudoperiferiálním vrcholem.

Asymptotická složitost algoritmu je $O(|E| + |V\ |log|V\ |)$. Jinými slovy, složitost algoritmu je vzhledem k nízkému počtu iterací rovna složitosti nalezení nejkratší cesty z vrcholu do ostatních vrcholů.

[3] dále alternativní pseudo-periferiální vrchol

2.6 Podmíněnost oblasti

Jedno z kritérií pro hodnocení oblasti se nazývá podmíněnost oblasti. Jedná se o kritérium založené na hraničních vrcholech.

Definice 2.18 *Vrchol $x \in V(G)$ se nazývá hraničním vrcholem oblasti G_i sítě G, jestliže je koncovým vrcholem vnější hrany, nebo je sousedním vrcholem vrcholu z jiné oblasti. Množina všech hraničních vrcholů oblasti G_i se značí $\partial V(G_i)$.*

Definice 2.19 *Nechť G je síť a G_i její oblast. Pak číslo podmíněnosti je definováno vztahem*

$$cond(G_i) = \min_{u \in V(G)} \frac{\max\limits_{v \in \partial V(G)} dist(u,v)}{\min\limits_{v \in \partial V(G)} dist(u,v)}.$$

Výhodou této podmíněnosti je jednoduchá názornost, nevýhodou časová složitost. Později si ukážeme jedno alternativní kritérium, které je rychlejší.

3 Rekurzivní grafová bisekce

3.1 Představení, podmínky a cíl RGB

RGB je založena na výběru oblasti a jejím následném rozdělení na dvě menší oblasti, což je prováděno několikrát za sebou. To dělá tuto metodu dělení grafu na oblasti rychlou, jednoduchou a efektivní. Podmínky na RGB jsou následující:

- Všechny oblasti vzniklé z RGB jsou souvislé.
- Oblasti jsou přibližně stejně velké, tj. jejich velikosti se od předepsané velikosti liší nejvíce o předem nastavenou toleranci.
- RGB se nezabývá podmíněností vzniklých oblastí.

Podmínka souvislosti je silnější než podmínka přibližně velkých oblastí. V případě, že není možné zároveň splnit obě podmínky, ustoupí se z podmínky přibližně stejně velkých komponent.

3.2 Předpočítání velikosti oblastí

Z důvodu lepší vyváženosti rozděleného grafu je zapotřebí, aby jednotlivé oblasti byly přibližně stejně velké. Proto je nezbytné před zahájením dělení grafu pomocí RGB předpočítat přibližnou velikost všech komponent, které vzniknou během dělení grafu rekurzivní grafovou bisekcí.

1. Vstupem je číslo n, které udává počet oblastí vzniklých dělením grafu na oblasti.
2. Inicializace: vytvoř nulový vektor $\mathbf{u}$ délky n - 1, který reprezentuje dělicí poměr vybrané oblasti grafu v daném kroku, a vektor $\mathbf{x} = (n, 0, \ldots, 0)$ délky n.
3. For $k = 1$ to $n - 1$ do
4. Vyber největší číslo ve vektoru $\mathbf{x}$ označ ho i.
5. Nahrad' i ve vektoru $\mathbf{x}$ číslem $\lfloor \frac{i}{2} \rfloor$ a označ ho j.
6. Na pozici $k+1$ ve vektoru $\mathbf{x}$ vlož číslo $i - j$.
7. Na pozici k vektoru $\mathbf{u}$ vlož číslo $\frac{j}{i}$.
8. End for.
9. Vektor $\mathbf{u}$ je výstupem.

Příklad 3.1
Mějme souvislý graf G určený k rozdělení na 5 oblastí. V inicializaci se vytvoří vektor $\mathbf{u} = (0;0;0;0)$ a vektor $\mathbf{x} = (5;0;0;0;0)$. Vektor $\mathbf{u}$ je vektor, ve kterém jsou uloženy jednotlivé dělicí poměry, složky vektoru $\mathbf{x}$ vyjadřují počet zástupných vrcholů. Zástupný vrchol je imaginární vrchol reprezentující všechny vrcholy patřící jedné oblasti.

V prvním kroku iterace z vektoru $\mathbf{x}$ vybereme nejvyšší hodnotu, tedy 5, a nastavíme $i = 5$. Dále vypočteme číslo $j = \lfloor \frac{i}{2} \rfloor = \lfloor \frac{5}{2} \rfloor = 2$ a upravíme hodnoty ve vektoru $\mathbf{x}$ na $(2;3;0;0;0)$. Na konci prvního kroku iterace vypočítáme první hodnotu vektoru $\mathbf{u}$, ta je $u(1) = \frac{j}{i} = \frac{2}{5} = 0,4$, a přejdeme k druhému kroku iterace.

Z vektoru $\mathbf{x}$ opět vybereme největší hodnotu, tou je číslo 3 na druhé pozici. Hodnotu proměnné i tedy změníme na 3 a číslo j změníme na $\lfloor \frac{i}{2} \rfloor = \lfloor \frac{3}{2} \rfloor = 1$. Vektor x je nyní $(2;1;2;0;0)$. V závěru druhého kroku iterace vypočítáme druhou hodnotu vektoru $\mathbf{u}$, $u(2) = \frac{j}{i} = \frac{1}{3} = 0,33$.

Další kroky už nebudeme podrobně rozepisovat. V třetím kroku iterace opět nejprve z vektoru $\mathbf{x}$ vybereme největší hodnotu, ta je na pozici 1 a nabývá hodnoty 2. Tedy po třetím kroku vektor $\mathbf{x}$ nabývá hodnot $(1;1;2;1;0)$ a vektor $\mathbf{u}$ je $(0,4;0,33;0,5;0)$.

Přejděme k poslednímu kroku iterace. Největší hodnota ve vektoru $\mathbf{x}$ je 2 a nachází se na třetí pozici. Po skončení této iterace je vektor $\mathbf{x}$ roven $(1;1;1;1;1)$ a vektor $\mathbf{u}$ nabývá hodnot $(0,4;0,33;0,5;0,5)$.

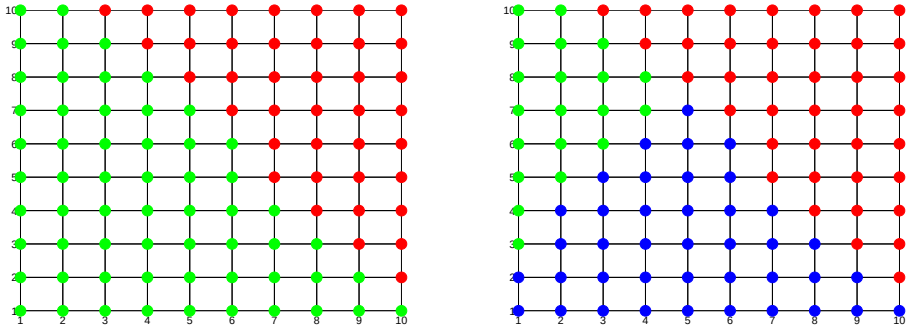
Vypočítaný dělicí poměr grafu G je $(0,4;0,33;0,5;0,5)$. ■

3.3 Algoritmus

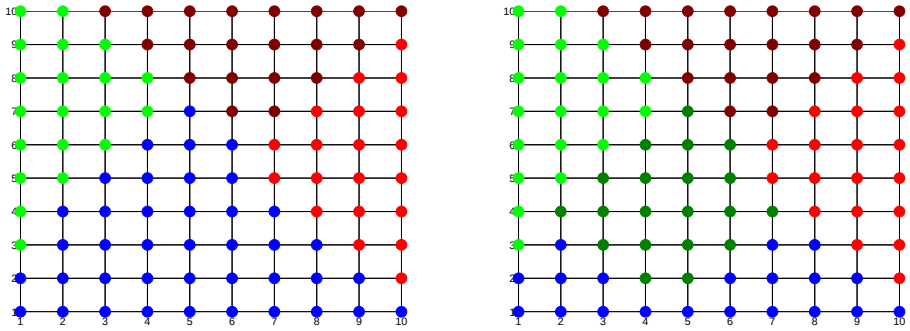
Algoritmus rekurzivní grafové bisekce zapsaný v pseudokódu:

1. Vstupem je graf G a číslo n, které udává počet komponent vzniklých dělením grafu na oblasti.
2. Inicializace: vytvoř množinu oblastí $M = \{G\}$ a nastav hodnotu k na 1.
3. While $k < n$ do
4. Vyber podgraf $H \in M : \forall F \in M : |V(F)| \leq |V(H)|$.
5. Nalezni v H pseudo-periferiální vrchol p a alternativní pseudo-periferiální vrchol q.
6. Nechť je l počet vrcholů, jež mají být přiřazeny menší z komponent.
7. Pomocí paralelního BFS se startovními vrcholy p a q vytvoř podgrafy H_1 a H_2. Tvoření komponent paralelním BFS ukonči, když $|V(H_i)| = l$; $i \in \{1,2\}$.
8. Vyber komponentu s menším počtem vrcholů a označ ji H_i. Druhou komponentu označ H_j.

Obrázek 1: Vlevo síť rozdělená na dvě oblasti, vpravo na tři oblasti.



Obrázek 2: Vlevo síť rozdělená na čtyři oblasti, vpravo na pět oblastí.



9. Pokračuj dále obyčejným BFS na komponentě M s prohledanou částí H_i.

10. Všechny doposud nepřiřazené vrcholy některé z komponent přiřaď komponentě H_j.

11. Jestliže $|V(H_j)| > (1 + tolerance) * l$, dovyvaž komponenty.

12. $M := M \setminus H$

13. $M := M \cup \{H_1, H_2\}$

14. $k = k + 1$

15. End while

Asymptotická složitost algoritmu je $O(n * (|E| + |V|log|V| + |V| + 2 * |E|))$.

Rozklad čtvercové diskretizace čtverce pomocí RGB je znázorněn na obrázcích 1 a 2.

Věta 3.1 *Nechť G je souvislý graf, $u, v \in V(G), u \neq v$. Nechť G_u a G_v jsou oblasti G vzniklé rozdělením G rekurzivní grafovou bisekcí. Pak tyto blasti G_u a G_v jsou souvislé.*

Důkaz. Nechť $\exists x \in G_u : dist_{G_u}(u, x) = \infty$. Ze souvislosti G plyne, že $dist_G(u, x) < \infty$. Proto všechny cesty z vrcholu u do vrcholu v obsahují alespoň jeden vrchol nepatřící G_u. Základem rekurzivní grafové bisekce je paralelní prohledávání do šířky. Algoritmus paralelní BFS pracuje následovně. V inicializaci vloží do fronty F_u vrchol u a do fronty F_v vloží vrchol v.

Dokud obě fronty nejsou prázdné, vyber frontu, jejíž první vrchol má menší vzdálenost, z ní vyber vrchol a vlož do fronty všechny sousední vrcholy posledně vybraného vrcholu, které ještě ve frontě nebyly. Všem právě vloženým vrcholům do fronty nastav vzdálenost o 1 větší než je vzdálenost posledně vybraného vrcholu.

Nyní dokažme korektnost jednotlivých částí:

- Vrchol k je přiřazen oblasti G_u pouze tehdy, je-li vybrán z fronty F_u.

  Jestliže byl vrchol b přiřazen oblasti G_u, byl v kroku před tímto přiřazením vybrán z fronty F_u. Musel být tedy do fronty F_u vložen, proto $\exists a \in V(G_u) : ab \in E(G)$ $AND\ dist_{G_u}(u, a) < \infty$. A proto $dist_{G_u}(u, b) < \infty$.

- Stejně jako 3.3, omezením paralelního prohledávání pouze jedním směrem se souvislost rozdělených oblastí neporuší.

- Tento krok je opět omezením paralelního prohledávání do šířky pouze jedním směrem, souvislost rozdělených oblastí se neporuší. Nyní se dopřiřadí všechny doposud nepřiřazené vrcholy, jinak by nastal spor s větou 2.2.

- Algoritmus dovyvažování vybere vrchol x a zkoumá, zda-li ho může přiřadit vedlejší komponentě, tedy zkoumá, zda-li se přeřazením vrcholu x do jiné oblasti neporuší souvislost obou oblastí.

■

3.4 Spektrální bisekce

Alternativní bisekcí je spektrální bisekce, ve které se využívají znalosti lineární algebry. Spektrální bisekce nalezne k zadanému grafu G Fiedlerův vektor.

Definice 3.1 *Laplaceova matice* $\mathbf{L}(\mathbf{G})$ *grafu G je čtvercová symetrická matice definována následovně:*

$$L(G)_{ij} = \begin{cases} deg(v_i) & pro\ \ i = j \\ -1 & pro\ \ i \neq j\ a\ e_{v_i v_j} \in E(G) \\ 0 & pro\ \ i \neq j\ a\ e_{v_i v_j} \notin E(G) \end{cases}$$

Laplaceova matice je pozitivně semidefinitní [15].

Definice 3.2 *Nechť* $\mathbf{A}$ *je pozitivně semidefinitní matice a* $\lambda_1,\ldots,\lambda_k$ *všechna nenulová vlastní čísla matice* A *taková, že* $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k > \lambda_{k+1} = 0$. *Číslo* λ_1 *se nazývá největší vlastní číslo,* λ_2 *se nazývá druhé největší vlastní číslo,* λ_{k-1} *se nazývá druhé nejmenší vlastní číslo a* λ_k *se nazývá nejmenší vlastní číslo.*

Definice 3.3 *Nechť je dán graf* G. *Vlastní vektor* $\mathbf{v}$ *Laplaceovy matice* $\mathbf{L}(\mathbf{G})$ *grafu* G *odpovídající druhému nejnižšímu vlastnímu číslu se nazývá Fiedlerův vektor.*

Algoritmus rozdělení grafu pomocí Fiedlerova vektoru:

```
1. Vypočti Fiedlerův vektor v matice L(G).
2. For each vrchol n
3.     If v_n < 0
4.         Vrchol n přidej do G_1.
5.     Else
6.         Vrchol n přidej do G_2.
7.     End if
8. End for
```

Následující věty vypovídají o důležitých vlastnostech spektrální bisekce.

Věta 3.2 *Nechť je dán graf* G. *Počet souvislých komponent grafu* G *je roven počtu nulových vlastních čísel* $\mathbf{L}(\mathbf{G})$.

Věta 3.3 *Nechť graf* G *je souvislý, pak podgraf* G_1 *je také souvislý. Jestliže* $\forall i \in \{1,\ldots,n\}$: $v(i) \neq 0$, *je podgraf* G_2 *také souvislý.*

Ve spektrální teorii se rovněž zavádí číslo podmíněnosti. Je jednoduché a efektivní.

Věta 3.4 *Číslo podmíněnosti grafu* G *je definováno jako* $cond(G) = \frac{\lambda_1}{\lambda_k}$.

4 Nalezení centra grafu

4.1 Definice

Řešení mnoha problémů vyžaduje zvolení zástupného vrcholu, který je blízko k ostatním vrcholům. Nejlépe vyhovující vrcholy se nazývají centra.

Definice 4.1 *Vrchol $v \in V(G)$ se nazývá centrem grafu G, jestliže $\forall u \in V(G) : ecc(u) \geq ecc(v)$.*

Níže popsaný algoritmus vybere vrcholy, které mohou být centry. Následně z nich vybere skutečná centra grafu.

Definice 4.2 *Vrchol $v \in V(G)$ se nazývá kandidátem na centrum grafu G s pseudo-periferiálními vrcholy p a q, jestliže $dist(p, v) + dist(q, v) = dist(p, q)$.*

4.2 Algoritmus založený na grafových algoritmech

1. Vstupem je graf G.
2. Nalezni pseudo-periferiální vrchol p.
3. Nalezni alternativní pseudo-periferiální vrchol q.
4. Nalezni všechny kandidáty na centrum a množinu všech kandidátů na centrum označ K.
5. $C = \{\ v \in K :\ ecc(v)\ =\ \max\{dist(p, v), dist(q, v)\}\}$
6. Množina C je množina vrcholů, které jsou centry grafu G.

Asymptotická složitost algoritmu je $O(|E| + |V|log|V|)$.

4.3 Spektrální metody nalezení centra grafu

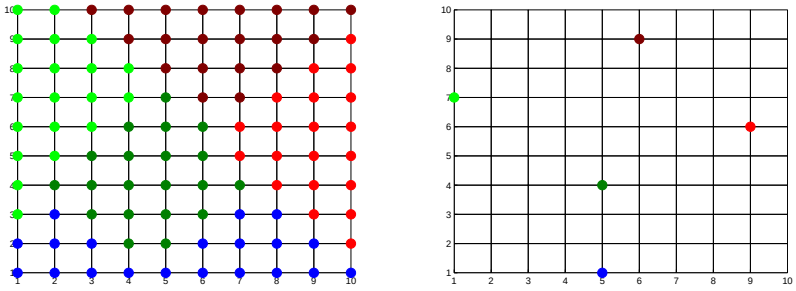
Následující text je převzat z [1]. Alternativním způsobem nalezení centra grafu jsou spektrální metody, které stojí na dobře známých teoriích z teorie grafů a lineární algebry. Graf je zde reprezentován maticí sousednosti.

Definice 4.3 *Matice sousednosti $\mathbf{A(G)}$ grafu G je čtvercová symetrická matice definována následovně:*

$$A(G)_{ij} = \begin{cases} 1 & \textit{pokud } i \neq j \textit{ a } e_{v_i v_j} \in E(G) \\ 0 & \textit{pokud } i = j \textit{ nebo } e_{v_i v_j} \notin E(G) \end{cases}$$

Věta 4.1 *Nechť D je matice sousednosti dané sítě a nechť $B = D^k$. Pak každé b_{ij} udává počet různých sledů v síti z vrcholu v_i do vrcholu v_j délky k.*

Obrázek 3: Vlevo síť rozložena na oblasti, vpravo centra jednotlivých oblastí



Je zřejmé, že je-li tato věta pravdivá, lze důkaz provést indukcí podle k.

Důkaz. Pro $k = 1$ je $B = D$, tedy matice B je maticí sousednosti. Jestliže je hrana ij součástí grafu, potom $d_{ij} = d_{ji} = 1$ a zde je pouze 1 cesta délky 1.

Předpokládejme, že pro nějaké $k \geq 1$ udává b_{ij} z $B = D^k$ počet (i, j)-cest délky k. Označme $D^{k+1} = C$, tedy $C = B * D$. Prvky matice C jsou tvaru:

$$c_{ij} = \sum_{l=1}^{n} b_{il} * d_{lj},$$

kde b_{il} udává počet cest z vrcholu v_i do vrcholu v_l délky k a d_{lj} udává počet cest z vrcholu v_l do vrcholu v_j délky 1 neboli, zda-li vede z vrcholu v_l do vrcholu v_j hrana. Jestliže graf neobsahuje hranu z vrcholu v_l do vrcholu v_j, pak je $d_{lj} = 0$ a $b_{il} * d_{lj} = 0$, tudíž zde není (i, j)-cesta délky $k+1$ s předposledním vrcholem l. Jestliže graf obsahuje hranu lj, je $d_{lj} = 1$, pak přidáme všechny (i, l)-cesty délky k. Jinak řečeno, spojením (i, l)-cesty délky k a hrany lj vznikne (i, j)-cesta délky $k + 1$. Protože všechny (i, j)-cesty mají jednoznačnou posloupnost postupně navštívených vrcholů, c_{ij} udává počet různých cest z vrcholu v_i do vrcholu v_j délky $k + 1$. ∎

Věta 4.2 *Nechť D je matice sousednosti dané sítě a $\mathbf{e} = [e_i], e_i = 1, i = 1, \ldots, n$. Číslo $w(i, k)$ počtu cest délky k se startovním vrcholem i je vyjádřeno následovně:*

$$w(i, k) = [\mathbf{D}^k e]_i.$$

Od sítě, která je přibližně regulární očekáváme, že více cest vznikne z vrcholu, který je blízko středu než z vrcholu blízko okraje sítě. Jednoduše řečeno vrchol i, který splňuje

$$w(i, k) \geq w(j, k), \forall j = 1, \ldots, n,$$

bude pro dostatečně velká k v okolí středu sítě.

Vektor

$$\mathbf{p} = \lim_{k \to \infty} \|D^k e\|^{-1} D^k e$$

je jednoznačný nezáporný vlastní vektor odpovídající největšímu vlastnímu číslu matice D, a je znám jako Perronův vektor matice D.

4.4 Lanczosova metoda

Lanczosova metoda pomocí řady transformací převede vstupní matici $\mathbf{A}$ na matici $\mathbf{B}$, jejíž vlastní čísla se vypočítají rychleji.

Inicializace této metody spočívá v nastavení vektorů $\mathbf{x_0} = \mathbf{y_0} = \mathbf{o}$. Vektory $\mathbf{x_1}$, $\mathbf{y_1}$ se zvolí náhodně tak, aby platilo ${\mathbf{x_1}}^T * \mathbf{y_1} \neq 0$.

Pomocí následujících rekurentních vzorců dostáváme posloupnost vektorů $\mathbf{x_k}$ a $\mathbf{y_k}$.

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} - b_k\mathbf{x_k} - c_{k-1}\mathbf{x_{k-1}}$$

$$\mathbf{y_{k+1}} = \mathbf{A^T}\mathbf{y_k} - b_k\mathbf{y_k} - c_{k-1}\mathbf{y_{k-1}},$$

kde čísla b_k a c_{k-1} jsou dána vztahy

$$b_k = \frac{\mathbf{y_k^T A x_k}}{\mathbf{y_k^T x_k}}, \qquad \text{pro } k = 1, \ldots, n-1$$

$$c_{k-1} = \frac{\mathbf{y_k^T x_k}}{\mathbf{y_{k-1}^T x_{k-1}}}, \qquad \text{pro } k = 2, \ldots, n-1 \text{ , } c_0 = 0.$$

Z důvodu korektnosti výpočtu je třeba zajistit $\forall k \in \{1, \ldots, n\} : \mathbf{y_k^T x_k} \neq \mathbf{0}$.
V případě, že $\exists j \in \{1, \ldots, n\} : \mathbf{y_j^T x_j} = \mathbf{0}$ výpočet algoritmu probíhá následovně:

1. $\mathbf{y_j} \neq \mathbf{0}$ a zároveň $\mathbf{x_j} \neq \mathbf{0}$. Je třeba začít znovu s jinou volbou vektorů $\mathbf{x_1}$ a $\mathbf{y_1}$.
2. $\mathbf{y_j} \neq \mathbf{0}$ a zároveň $\mathbf{x_j} = \mathbf{0}$. Vem za $\mathbf{x_j}$ libovolný vektor, jenž je ortogonální ke všem vektorům $\mathbf{y_1}, \ldots, \mathbf{y_{j-1}}$. V druhé rovnici polož $c_{j-1} = 0$.
3. $\mathbf{y_j} = \mathbf{0}$ a zároveň $\mathbf{x_j} \neq \mathbf{0}$. Vem za $\mathbf{y_j}$ libovolný vektor, jenž je ortogonální ke všem vektorům $x_1, \ldots, x_{j-1}$. V první rovnici polož $c_{j-1} = 0$.
4. $\mathbf{y_j} = \mathbf{0}$ a zároveň $\mathbf{x_j} = \mathbf{0}$. V obou rovnicích polož $c_{j-1} = 0$.

Zavedeme-li $\mathbf{S} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]$, je tato matice regulární a platí

$$\mathbf{S^{-1}AS} = \mathbf{B},$$

kde

$$\mathbf{B} = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ 1 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & 1 & b_n \end{bmatrix}.$$

Výše uvedené vztahy převádí zadanou matici na podobnou tři-diagonální matici. Vlastní čísla matice lze spočítat některou ze známých metod výpočtu vlastních čísel, které

jsou uvedeny například v [13]. Vztah mezi vlastním vektorem $\mathbf{e}$ matice $\mathbf{A}$ a vlastním vektorem $\mathbf{f}$ matice $\mathbf{B}$ odpovídající témuž vlastnímu číslu je dán vztahem

$$\mathbf{e} = \mathbf{S}\mathbf{f}.$$

Lanczosova metoda je citlivá na zaokrouhlovací chyby, proto v následující podkapitole je popsaná její modifikace, která je numericky více stabilní.

4.5 Modifikovaná Lanczosova metoda

Inicializace je stejná jako v předchozím případě, opět nastavíme $\mathbf{x_0} = \mathbf{y_0} = \mathbf{o}$ a vektory $\mathbf{x_1}$, $\mathbf{y_1}$ zvolíme náhodně tak, aby platilo ${\mathbf{x_1}}^T * \mathbf{y_1} \neq 0$.

Pomocí následujících rekurentních vzorců dostáváme posloupnost vektorů $\mathbf{x_k}$ a $\mathbf{y_k}$.

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} - \sum_{i=1}^{k} c_{ki}\mathbf{x_i}$$

$$\mathbf{y_{k+1}} = \mathbf{A^T}\mathbf{y_k} - \sum_{i=1}^{k} d_{ki}\mathbf{y_i},$$

kde čísla c_{ki} a d_{ki} jsou dána vztahy

$$c_{ki} = \frac{\mathbf{y_i^T}\mathbf{A}\mathbf{x_k}}{\mathbf{y_j^T}\mathbf{x_j}},$$

$$d_{ki} = \frac{\mathbf{y_i^T}\mathbf{A^T}\mathbf{x_k}}{\mathbf{y_i^T}\mathbf{x_k}},$$

pro $i = 1, \ldots, k$.

Z důvodu korektnosti výpočtu je třeba zajistit $\forall k \in \{1, \ldots, n\} : \mathbf{y_k^T}\mathbf{x_k} \neq \mathbf{0}$.

V případě, že $\exists j \in \{1, \ldots, n\} : \mathbf{y_j^T}\mathbf{x_j} = \mathbf{0}$, probíhá výpočet algoritmu úplně stejně jako u Lanczosovy metody.

Zavedeme-li $\mathbf{S} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]$, je tato matice regulární a platí

$$\mathbf{S^{-1}}\mathbf{A}\mathbf{S} = \mathbf{B},$$

kde

$$\mathbf{B} = \begin{bmatrix} c_{11} & c_{21} & \cdots & \cdots & c_{n1} \\ 1 & c_{22} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & c_{nn-1} \\ 0 & \cdots & 0 & 1 & c_{nn} \end{bmatrix}.$$

Výše uvedené vztahy převádí zadanou matici na podobnou matici v Hessenbergově tvaru. Vlastní čísla této matice lze spočítat některou ze známých metod výpočtu vlastních

čísel, které jsou vhodné pro výpočet vlastních čísel matic v Hessenbergově tvaru. Vztah mezi vlastním vektorem $\mathbf{e}$ matice $\mathbf{A}$ a vlastním vektorem $\mathbf{f}$ matice $\mathbf{B}$ odpovídající témuž vlastnímu číslu je dán vztahem

$$\mathbf{e} = \mathbf{S}\mathbf{f}.$$

5 Vícevrcholová nastavovací expanze

5.1 Představení, podmínky a cíl MLSE

Vícevrcholová nastavovací expanze je úprava oblastí vzniklých rozdělením grafu na oblasti, která sníží podmíněnost jednotlivých oblastí. Metoda přímo nerozděluje graf na oblasti, jen přerozdělí vrcholy jednotlivým oblastem. Algoritmus vychází ze zadaných středů jednotlivých oblastí, ze kterých vychází při přiřazovaní ostatních vrcholů. Výsledek lze zlepšit odstraněním různých defektů. Jeden z defektů a jeho odstranění si nyní představíme.

5.2 Vyhlazovací algoritmy

Je-li to možné, přiřadí vyhlazovací algoritmy vrcholy defektu jiné oblasti, ve které tyto vrcholy defekt netvoří. Výhodou odstranění defektů je lepší podmíněnost jednotlivých oblastí.

Definice 5.1 *Nechť je dán souvislý graf G a souvislá oblast G_u, která je obsažena v grafu G. Podgraf P_{G_u} oblasti G_u se nazývá defektem, jestliže*

- $V(P_{G_u}) \in V(G_u)$,
- $\forall u, v \in V(P_{G_u}) : dist_{G_u}(u, v) < \infty$,
- $\exists! u \in V(P_{G_u}) : deg_{G_u}(u) = 1$,
- $\forall v \in V(P_{G_u}) : deg_{G_u}(v) \leq 2$.

Definice 5.2 *Nechť je dán souvislý graf G a souvislá oblast G_u, která je obsažena v grafu G. Nechť oblast G_u obsahuje defekt P_{G_u}. Vrchol $u \in V(P_{G_u})$ se nazývá koncovým vrcholem defektu P_{G_u}, jestliže $deg_{G_u}(u) = 1$.*

Rozdělení defektu na vnější a vnitřní dobře poslouží při jeho odstraňování.

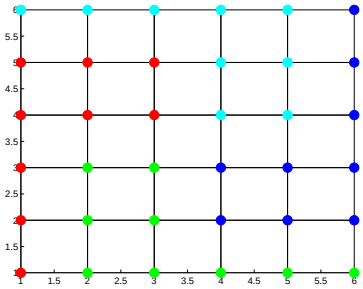
Definice 5.3 *Nechť je dán souvislý graf G a souvislá oblast G_u, která je obsažena v grafu G. Nechť oblast G_u obsahuje defekt P_{G_u}. Vrchol $v \in V(P_{G_u})$ se nazývá vnitřním vrcholem defektu, jestliže není koncovým vrcholem defektu.*

Definice 5.4 *Nechť je dán souvislý graf G a souvislá oblast G_u, která je obsažena v grafu G. Defekt P_{G_u} se nazývá vnitřním defektem, jestliže $\forall u \in V(P_{G_u}) : deg_{G_u}(u) < deg_G(u)$.*

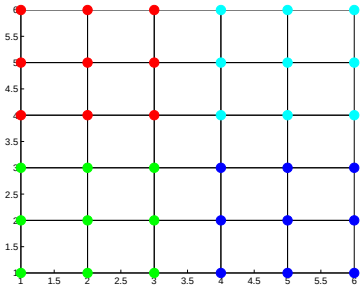
Definice 5.5 *Nechť je dán souvislý graf G a souvislá oblast G_u, která je obsažena v grafu G. Defekt P_{G_u} se nazývá vnějším defektem, jestliže $\forall u \in V(P_{G_u}) : deg_{G_u}(u) = deg_G(u)$.*

Z důvodu větší efektivity hledání a následného odstranění defektů je výhodné využít heuristiku. Zde je představena heuristika založená na nalezení optimální kostry a následném přeřazení všech vhodně přeřaditelných vrcholů. Kostra je souvislý acyklický podgraf na všech vrcholech grafu G. Tato metoda vyhlazení se zaměřuje pouze na vhodné vrcholy k vyhlazení, nikoliv na všechny.

Obrázek 4: Síť rozdělená na oblasti, které mají defekt.



Obrázek 5: Síť rozdělená na oblasti s odstraněnými defekty.



Definice 5.6 *Nechť je dán souvislý graf G, kostra T grafu G se nazývá optimální, jestliže splňuje následující podmínky:*

- $|\{u \in V(G) : deg_T(u) = 1\}|$ *je minimální.*
- *Počet vrcholů, které jsou součástí některého z defektu, je minimální.*

5.2.1 Vyhlazení prohledáváním do hloubky

Algoritmus stojí na postupném procházení vrcholů oblastí a u každého zkoumá, zda-li ho má přeřadit do jiné oblasti nebo nikoliv. Vrchol se přeřadí do jiné oblasti právě tehdy, když je koncovým vrcholem defektu v současné oblasti a zároveň by v nové oblasti nebyl součástí defektu.

1. Vstupem je graf G a oblasti $G_1, \ldots, G_n$.
2. For each oblast $G_1, \ldots, G_n$
3. Pomocí DFS s náhodným počátečním vrcholem postupně zkontroluj podgraf G_i.

4. Kontrola jednoho vrcholu v probíhá následovně:

5. Zkontroluj vrchol v, zdali má být přeřazen do jiné oblasti, a případně ho přeřad'.

6. If existuje nějaký potomek vrcholu u, který ještě nebyl navštíven, navštiv ho a pokračuj bodem číslo 5 pro vrchol u. Po ukončení práce s vrcholem u, znovu zkontroluj vrchol v.

7. End if

8. End for

5.2.2 Vyhlazení optimální kostrou

Algoritmus nalezne optimální kostru a poté zkoumá její defekty. Jestliže jsou defekty také v původním grafu, zkoumá, zda-li je má přeřadit do jiné oblasti nebo ne. Vrchol se přeřadí do jiné oblasti právě tehdy, když je koncovým vrcholem defektu v současné oblasti a zároveň by v nové oblasti nebyl součástí defektu.

Algoritmus nalezení optimální kostry:

1. Vstupem je souvislý graf G.

2. Vyber náhodný vrchol $x \in V(G)$.

3. Vytvoř $S = V(G) \setminus \{x\}$.

4. Vytvoř $T; V(T) = \{x\}$.

5. While $S \neq \emptyset$

6. Vyber jeden z vrcholů v se stavem nezkoumán, pro který $deg_{T_i}(v) = 1$ a označ ho u.

7. Jestliže je vrchol u v podgrafu G_i koncovým vrcholem vnitřního defektu, zjisti, zda by v jiné oblasti nebyl součástí defektu.

8. End while

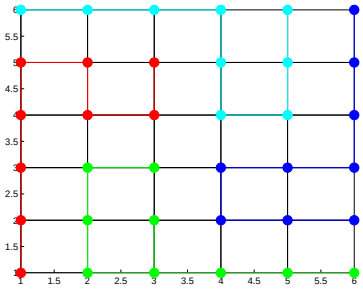
Nyní, když víme jak nalézt optimální kostru, můžeme přejít k vyhlazení optimální kostrou.

1. Vstupem je graf G a oblasti $G_1, \ldots, G_n$.

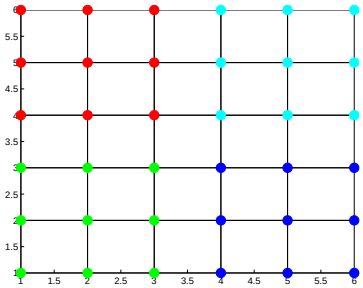
2. For each oblast $G_1, \ldots, G_n$

3. Nalezni optimální kostru T_i oblasti G_i.

Obrázek 6: Síť rozdělená na oblasti s defekty a nalezenými kostrami.

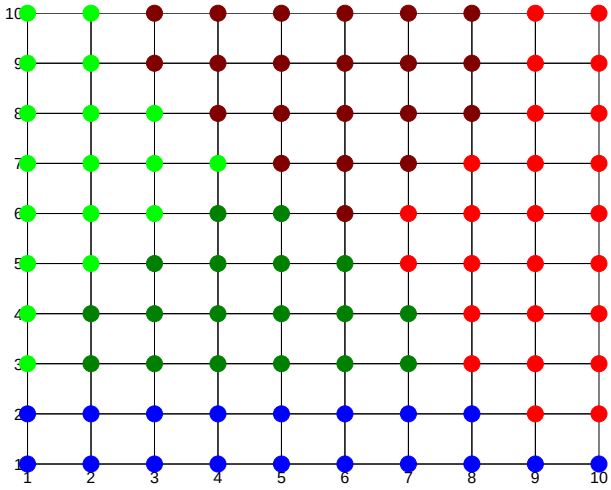


Obrázek 7: Síť rozdělená na oblasti s odstraněnými defekty.



4. Všem vrcholům oblasti G_i nastav stav na nezkoumán.
5. While existuje vrchol v se stavem nezkoumán, pro který $deg_{T_i}(v) = 1$:
6. Vyber jeden z vrcholů v se stavem nezkoumán, pro který $deg_{T_i}(v) = 1$ a označ ho u.
7. Jestliže je u v podgrafu G_i koncovým vrcholem vnitřního defektu, zjisti, zda by v jiné oblasti nebyl součástí defektu.
8. End while
9. End for
10. Výstupem jsou nové oblasti oblasti grafu $G_1, \ldots, G_n$.

Obrázek 8: Síť vyhlazena MLSE



5.3 Algoritmus vícevrcholové nastavovací expanze

Teď když známe metodu na odstranění jednotlivých defektů, vraťme se zpátky k vícevrcholové nastavovací expanzi.

1. Vstupem je graf G a středy zadaných oblastí $c_1, \ldots, c_n$.
2. Výchozími vrcholy jsou jednotlivé středy zadaných oblastí.
3. Paralelním BFS přiřaď doposud nepřiřazené vrcholy některé z oblastí.
4. Odstraň všechny odstranitelné defekty jednotlivých oblastí.
5. Výstupem jsou oblasti grafu $G_1, \ldots, G_n$.

6 Metody rozdělení grafu založené na shlukování vrcholů

6.1 Představení

Následující kapitola se věnuje algoritmům založeným na shlukování vrcholů, které pro rozdělení sítě využívá například program Metis. Metis je software pro rozdělení velkých sítí a výpočet redukujícího přeuspořádání řídké matice. Algoritmus je založen na víceúrovňové grafové bisekci.

Tradiční algoritmy dělí síť přímo na původní síti, což je velmi často pomalé a neefektivní. Víceúrovňové algoritmy rozkládají síť jinak. Tyto algoritmy redukují velikost sítě, následně rozdělí malou síť a toto rozdělení převedou zpátky na původní síť.

Tento program umí během několika vteřin rozdělit síť s více než miliónem vrcholů na 256 oblastí[4].

6.2 Víceúrovňová grafová bisekce

Myšlenka víceúrovňové grafové bisekce je jednoduchá. Nejdříve převedeme vstupní síť na menší síť, tu rozdělíme a pak toto rozdělení transformujeme zpět na původní síť. Fáze víceúrovňové grafové bisekce jsou následující:

- **Shlukovací fáze:** Vstupní síť G_0 je postupně převedena na posloupnost menších sítí G_1,...,G_n takových, že platí $|V(G_0)| \geq |V(G_1)| \geq \ldots \geq |V(G_n)|$.
- **Rozdělovací fáze:** V síti G_n se nalezne rozdělení R_n, jehož každá oblast obsahuje přibližně stejný počet vrcholů původní sítě G_0.
- **Zpětná fáze:** Rozdělení R_n sítě G_n se převede zpátky na rozdělení R_0 sítě G_0 pomocí částečných projekcí R_{n-1},R_{n-2},...,R_1.

6.3 Shlukovací fáze

Shlukovací fáze vyrobí k vstupní síti G projekce Q_1,...,Q_{n-1} a odpovídající sítě G_1,...,G_n menší velikosti. Je vhodné, aby každý vrchol sítě G_i reprezentoval přibližně stejný počet vrcholů sítě G_0, a proto se zde dobře uplatní párování. Párováním se neporuší souvislost indukovaných podgrafů reprezentovaných jedním vrcholem sítě G_i.

Definice 6.1 *Množina hran M se nazývá párování, jestliže $\forall u, v \in M : u \cap v = \emptyset$.*

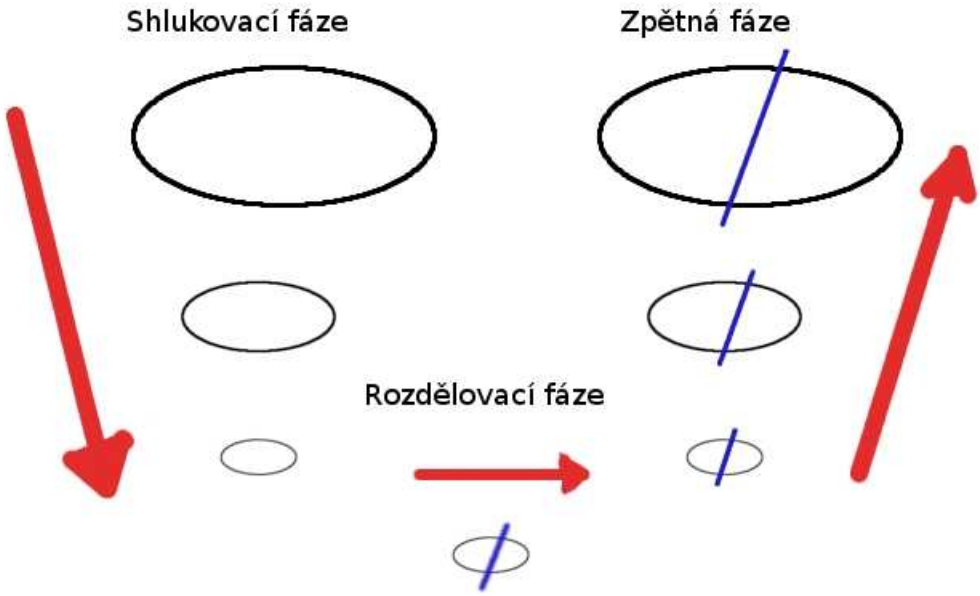
Z důvodu efektivnosti shlukování jsou výhodná největší a maximální párování.

Definice 6.2 *Párování M je největší, jestliže pro každé párování N grafu G platí $|M| \geq |N|$.*

Definice 6.3 *Párování M se nazývá maximální, jestliže pro každé párování N grafu G takové, že $M \subseteq N$, platí $M = N$.*

[4] zdroj [14]

Obrázek 9: Schéma víceúrovňové grafové bisekce.



Projekce Q_i reprezentuje spárování vrcholů sítě G_{i-1}, pomocí kterého vznikne síť G_i.

Rozměr nové sítě silně závisí na počtu hran v párování. Větší počet hran v párování vede k menší síti. Proto je výhodné zvolit maximální párování.

První způsob výběru párování je náhodné párování. Náhodně se vybere nějaké maximální párování, vrcholy se sloučí a aktualizují se hrany včetně jejich vah. Výhodou je jednoduchost a rychlost, nevýhodou určitá náhodnost výsledku.

Další způsob výběru párování se nazývá největší hranové párování. Vybere se z maximálních párování největší, vrcholy se sloučí a aktualizují se hrany včetně jejich vah. Výhodou je efektivnost a minimalizace vah hran nově vytvořeného grafu.

Shlukování pomocí největšího hranového párování nachází uplatnění hlavně při paralelní a distribuované implementaci dekompozice, kde kvalitu řešení ovlivňuje velikost řezu neboli počet hran mezi sousedními oblastmi. Velikost řezu vzhledem ke kvalitě řešení není významně důležitá pro sekvenční implementaci dekompozice. Algoritmus největšího hranového párování je uveden v [10]. Nyní si představíme jednoduchý algoritmus pro nalezení maximálního párování převzatý z [3].

Pro následující použití zaveďme operaci symetrický rozdíl.

Definice 6.4 *Symetrický rozdíl je množina hran* $\{e : e \in M \; nebo \; e \in P; e \notin M \cap P\}$.

Důležitým pojmem je volný vrchol. Maximální párování obsahuje ze všech párování nejnižší počet volných vrcholů.

Definice 6.5 *Nechť je dán graf $G = (V, E)$ a jeho párování M. Vrchol $v \in V(G)$ se nazývá volný, jestliže není koncovým vrcholem libovolné hrany patřící párování M.*

V části teorie grafů věnované párování se často vyskytují pojmy alternující a rozšiřující cesta. Obě tyto cesty se uplatňují při zjišťování, zda-li lze párování zvětšit, tj. nalézt nové párování v grafu G s větším počtem hran.

Definice 6.6 *Nechť je dán graf $G = (V, E)$ a jeho párování M. Cesta P se nazývá alternující, jestliže její hrany střídavě patří a nepatří párovaní M.*

Definice 6.7 *Nechť je dán graf $G = (V, E)$ a jeho párování M. Cesta P se nazývá rozšířující, jestliže je alternující cestou mezi dvěma volnými vrcholy.*

Následující algoritmus nalezení rozšiřující cesty je převzat z [17]. Vstupem je graf G a jeho párování M, výstupem je rozšiřující cesta, která je v případě nenalezení prázdná.

1. Vstupem je graf G a jeho párování M.
2. $F := \emptyset$.
3. Odznač všechny vrcholy a hrany grafu G.
4. Označ všechny hrany párování M.
5. For each volný vrchol v
6. Vytvoř strom T, $V(T) = \{v\}$ a vlož ho do množiny F
7. End for
8. While když je v F neoznačený vrchol v se sudou vzdáleností od kořene stromu:
9. While $\exists w \in V(G)$, že hrana vw je neoznačená:
10. If $v \notin F$,
11. $x \in V(G) : xw \in M$,
12. $F := F \cup \{vw, wx\}$.
13. Označ hranu wx.
14. Else
15. If vzdálenost vrcholu w od kořene je sudá.
16. If kořeny vrcholů v a w jsou různé.

```
17.            Označ P cestu z kořene stromu s vrcholem v přes
               vrcholy v a w do kořene stromu s vrcholem w.
18.            Vrať cestu P.
19.          Else
20.            Označ B květ vytvořený z hrany vw a cesty vw
               z T.
21.            Zúžením G a M přes B vznikne graf G' a párování
               M'.
22.            Nalezni grafu G' a párováním M' rozšiřující cestu
               P' .
23.            Převeď nalezenou rozšiřující cestu P' grafu G'
               na rozšiřující cestu P grafu G.
24.            Vrať cestu P.
25.          End if
26.        End if
27.      End if
28.      Označ hranu vw.
29.    End while
30.    Označ vrchol v.
31. End while
32. Vrať prázdnou cestu.
```

Věta 6.1 *Párování M není maximální právě tehdy, když existuje rozšiřující cesta vzhledem k M.*

Důkaz. ($\Rightarrow$) Jestliže je P rozšiřující cesta vzhledem k párování M, pak symetrický rozdíl $M \bigoplus P$ je párování a $|M \bigoplus P| = |M| + 1$.

($\Leftarrow$) Nechť je M jiné než maximální párování a N je maximální párování. Uvažujme symetrickou diferenci párování $M \bigoplus N$. Protože neexistují dvě hrany v párování se společným vrcholem, každý vrchol má nejvýše jeden sousední vrchol v párování M a nejvýše jeden sousední vrchol v párování N. Každá souvislá komponenta z $M \bigoplus N$ je jedním z následujících typů:

1. Cesta s první a poslední hranou patřící párování M.
2. Cesta s první a poslední hranou patřící párování N.

3. Cesta s první hranou patřící párování M a poslední hranou patřící párování N.
4. Střídavá kružnice, jejíž hrany střídavě patří párování M a N.

Jelikož N je maximální párování a M ne, víme, že $|N| > |M|$. Proto alespoň jedna komponenta obsahuje více hran z párování N než M. Také střídavé kružnice musí být sudé délky, protože každý vrchol této kružnice je incidentní s nejvýše jednou hranou párování N a jednou hranou z párování M. Proto existuje komponenta typu 2 – rozšiřující cesta. ■

Důsledkem této věty je následující algoritmus.

1. Vstupem je graf G.
2. Nastav párování M grafu G na párování neobsahující hrany.
3. While v grafu G pro párování M existuje rozšiřující cesta.
4. Nechť P je rozšiřující cesta.
5. $M := M \bigoplus P$
6. End while

Tento algoritmus je znám jako Edmondsův algoritmus a jeho složitost je $O(mn^2)$, kde n je počet vrcholů grafu a m počet hran grafu.

6.4 Zpětná fáze

Protože rozdělovací fáze je popsána v předchozích kapitolách, přejděme rovnou k zpětné fázi. Cílem zpětné fáze je převést rozdělení R_n sítě G_n na rozdělení R_0 původní sítě G_0. Rozdělení R_0 sítě G lze snadno zkonstruovat pomocí projekce Q_0 a rozdělení R_1 sítě G_1, které známe.

Algoritmus zpětné fáze zapsaný v pseudokódu:

1. Vstupem jsou sítě $G_0,\ldots,G_n$, projekce $Q_1,\ldots,Q_n$ a rozdělení sítě P_n.
2. For $i := n-1,\ldots,1$
3. For $\forall u \in V(G_i)$
4. Nalezni vrchol $v \in V(G_{i+1}) : v = Q_{i+1}(u)$.
5. $R_i(u) := R_{i+1}(v)$.
6. End for
7. End for

Jiné algoritmy zpětné fáze jsou popsány v [7].

7 Aplikace na řešení inženýrských úloh

7.1 Vyhodnocení

Měření proběhlo na čtvercové síti 10×10, která byla rozdělena na 5, 6 a 8 oblastí metodou rekurzivní grafové bisekce[5], rekurzivní grafové bisekce s vyhlazováním[6] a spektrální bisekce[7]. V tabulce 7.1 kritérium 1 znamená podmíněnost podle kritéria založeného na hranici oblasti, zatímco kritérium 2 značí spektrální kritérium.

Pár poznámek k hodnotám z tabulky 7.1: pro obě kritéria platí, že menší hodnota je lepší. Zajímavé je srovnání mezi vyhlazenými oblastmi a nevyhlazenými, kde se závěry kritérií různí. První kritérium lépe vyznívá pro vyhlazené oblasti, zatímco druhé pro nevyhlazené.

Srovnání jednotlivých metod vyznívá nejlépe pro vyhlazené metody, nevyhlazené a spektrální jsou na tom přibližně stejně. Zajímavé jsou výsledky podle počtu oblastí. Z výsledku kritéria 2 vyplývá, že menší počet oblastí je lepší. Medián není vypovídající k hodnocení dekompozice sítě, protože vyhodnocení podle prvního kritéria se téměř neliší, zatímco ve vyhodnocení podle druhého kritéria jsou rozdíly hodnot malé.

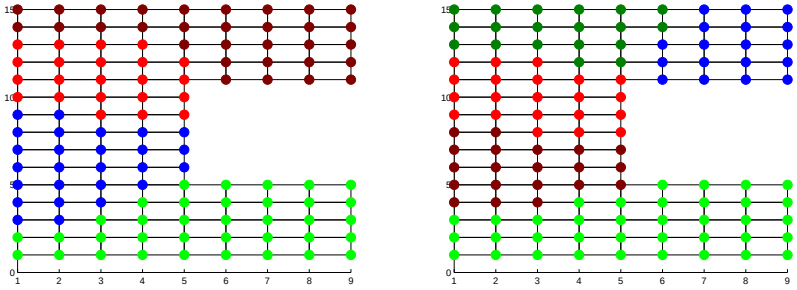
7.2 Výsledky praktických úloh

[5] V 7.1 označeno nevyhlazené.
[6] V 7.1 označeno vyhlazené.
[7] V 7.1 označeno spektrální.

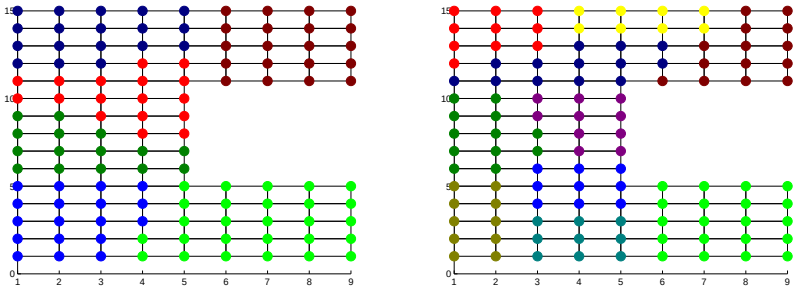
Tabulka 1: Vyhodnocení rozkladu čtvercové sítě

rozdělení	oblasti	Kritérium 1			Kritérium 2		
		maximum	průměr	medián	maximum	průměr	medián
nevyhlazené	5	6.0000	4.2000	4.0000	1.7407	1.6796	1.6951
	6	6.0000	4.3333	4.0000	2.1380	1.9005	1.8918
	8	4.0000	3.1250	3.0000	5.2814	2.4779	2.0507
vyhlazené	5	4.0000	3.8000	4.0000	2.0099	1.7263	1.6928
	6	5.0000	3.6667	3.0000	2.5009	1.9563	1.8291
	8	5.0000	3.1250	3.0000	8.4852	2.9478	2.0740
spektrální	5	6.0000	4.6000	4.0000	1.9993	1.6926	1.5734
	6	5	4	4	4.2603	2.4091	2.1166
	8	5.0000	3.5000	3.0000	7.8864	3.3003	1.9693

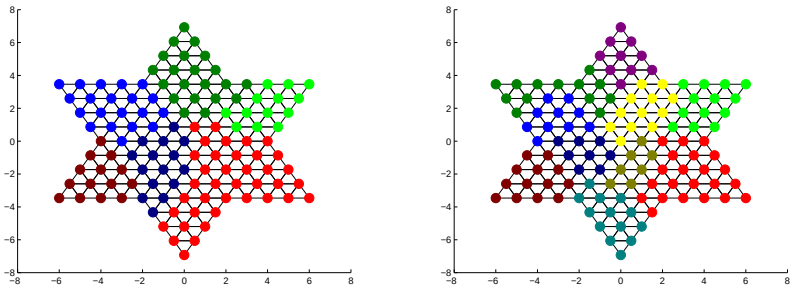
Obrázek 10: C rozdělené na 4 a 5 oblastí



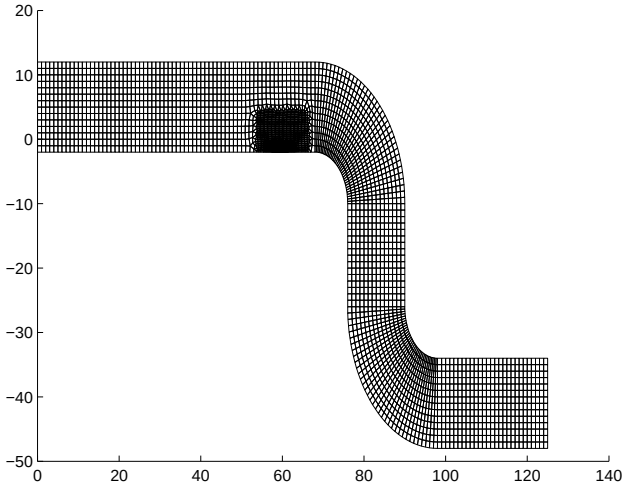
Obrázek 11: C rozdělené na 6 a 10 oblastí



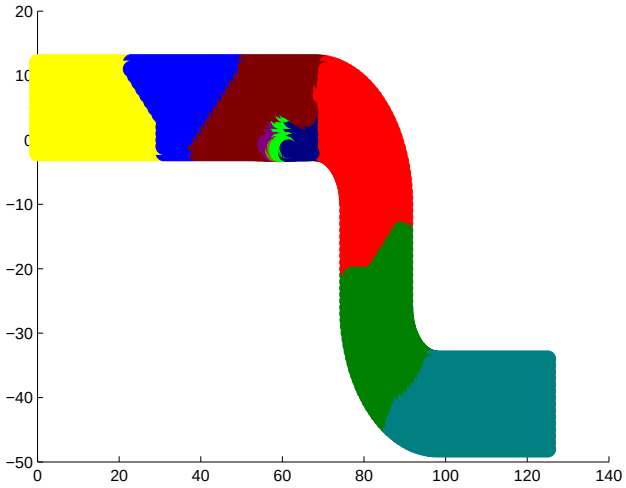
Obrázek 12: Hvězda rozdělená na 6 a 10 oblastí



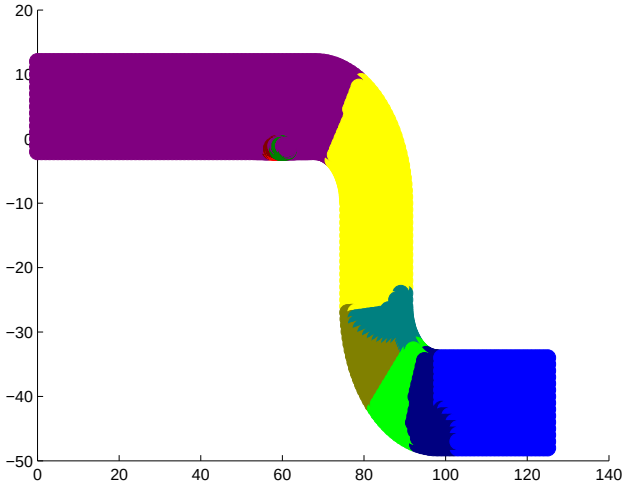
Obrázek 13: Výztuž



Obrázek 14: Výztuž rozdělená na 10 oblastí



Obrázek 15: Výztuž rozdělená spekrální bisekci na 10 oblastí



8 Závěr

V této práci jsem představil různé způsoby rozdělení souvislého grafu na oblasti, především metody založené na grafových algoritmech a metody stojící na spektrální teorii.

Zvolený postup rozdělení grafu spočíval v rozdělení grafu pomocí grafové bisekce a vyhledání center nalezených oblastí. Z důvodu zlepšení podmíněnosti oblastí jsme následně pomocí vícevrcholové nastavovací expanze přeřadili některé vrcholy z jednotlivých oblastí do jiných. Další optimalizací rozkladu bylo odstranění defektu jednotlivých oblastí.

Kromě výkladu teorie jsem implementoval řešení dekompozice grafů v Matlabu, které jsem nasadil na řešení vybraných inženýrských úloh – viz příloha.

9 Reference

[1] Brzobohatý T., Dostál Z., Kozubek T., Kovář P., Makropulos A., *Cholesky-SVD decomposition with fixing nodes to stable computation of a generalized inverse of the stiffness matrix of a floating structure*, submitted to Int. J. Numer. Meth. Engng 2010.

[2] Demmel J., *CS 267: Notes for Lecture 23, April 9, 1999. Graph Partitioning, Part 2.*, `http://www.cs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html`.

[3] Karp R., Edmonds's Non-Bipartite Matching Algorithm, Course Notes. U. C. Berkeley

[4] Karypis G., Kumar V., *A fast and highly quality multilevel scheme for partitioning irregular graphs.*, SIAM Journal on Scientific Computing, 1998 (to appear). Also available on WWW at `http://www.cs.umn.edu/~karypis`. A short version appears in Intl. Conf. on Parallel Processing 1995.

[5] Karypis G., Kumar V.,*Metis* A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices,Version 4.0*

[6] Karypis G., Kumar V., *Multilevel algorithms for multi-constraint graph partitioning.*, Technical Report TR 98-019, Department of Computer Science, University of Minnesota, 1998.

[7] Karypis G., Kumar V., *Multilevel graph partitioning schemes*, Proc. Int. Conf. on Parallel Processing, 1995

[8] Karypis G., Kumar V., *Multilevel k-way partitioning scheme for irregular graphs.*, J. Parallel Distrib. Comput. 48(1): 96-129, 1998. Computing, 48(1):96–129, 1998. Also available on WWW at `http://www.cs.umn.edu/~karypis`

[9] Kovář P., *Teorie grafů*, 2009

[10] Lovász L., Plummer M., *Matching Theory*, Akadémiai Kiadó, 1986

[11] Matoušek J., Nešetřil J., *Kapitoly z diskrétní matematiky*, Praha, Nakladatelství Karolinum, 2003

[12] Saad Y.,*Iterative Methods for Sparse Linear Systems*, SIAM, 2003

[13] Vitásek E., *Numerické metody*, SNTL, 1987

[14] `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`

[15] `http://martin.lipinsky.cz/skola/dma/prednasky/7.pdf`

[16] `http://cs.wikipedia.org/wiki/Rovinn\%C3\%BD\_graf`

[17] http://en.wikipedia.org/wiki/Edmonds\%27s_matching\
_algorithm

A Přiložené matlabovské soubory

adj2lapl.m
adj2negh.m
center.m
dist.m
dist1.m
hlavni.m
hlavnis.m
hvykresli.m
hvykreslimst.m
hresize.m
lapl2adj.m
lapl2negh.m
mlse.m
nacti_vyztuz.m
negh2adj.m
negh2lapl.m
paleta.m
ppnote.m
rgb1.m
sit1PET.m
sit2PET.m
sit4PET.m
sit5PET.m
sit6PET.m
sit7PET.m
sit8PET.m
sit9PET.m
sizes.m
spektb.m
spekt.m
vyhlazeni1.m
vyhlazeni2.m
vykresli.m